CDS 130 Computing for Scientists
Final exam review

Topics:

MATLAB

1. Variables
   1) Scalars, vectors, matrices
   2) Assignment

2. Create vectors : Row vector and Column vector
   method 1: A(5) = 1;
   method 2: A = [2,3,4,5]; or A=[2; 3; 4; 5;]
   For row vectors, you can use either commas or spaces to separate the elements
   method 3: A = 2: -1: -6; (colon notation).
   If the increment is 1, the short-handed notation is A = 2:5; (i.e., A=[2, 3, 4, 5])
   method 4: form a new vector from previously defined vectors
   >> A = [1, 2, 3];
   >> B = [3, 4, 5];
   >> C = [A, B]
   method 5: iterations
   A(1) = 0;
   for i = 2:6
   A(i) = i
   end
   % this creates a new vector [0, 2, 3, 4, 5, 6], and it does so by appending new pieces onto an existing vector, one piece at a time.
   method 6: A = rand(5) % this creats a vector with all element values between 0 and 1.

   * Not tested: linspace(), zeros() and ones().

3. Vectors in Matrix
   1) row vector: A(5) means A(1, 5)
   2) column vector A(5,1)
   3) Transpose operator: A(5)' is now a column vector

4. Address vector elements
   1) meaning of A(1,5) and A(5)
   2) meaning of A(5,1)
   3) A(1:3) is a new vector
   4) A(0.2) is invalid

5. Vector operations
   1) Vector added by a scalar
   >> A = [1, 2, 3];
   >> B = A + 3.0
   B =

4 5 6

3) Vector multiplied by a scalar

```
>> A = [1, 2, 3];
>> B = A* 3.0;
 B =
    3   6   9
```

2) Add two vectors

```
>> A = [1, 2, 3]
>> B = [3, 4, 5 ]
>> C = A+B
 C =
  [4, 6, 8]
```

4) Vector summation (sum of all vector elements)

```
>> C = [4, 6, 8];
>> sum(C)
 ans = 20
```

5) Element-by-element operation (Attach . to the first vector) (extra credit)

```
>> C = A. * B
 C =
    [3, 8, 15]

>> C = [1 1 1]. / A;
 C =
    1   0.5  0.33333
```

6. Create Matrices

method 1: A(4,5) = 0.0;
method 2: A = [1, 2, 3; 3, 4, 5];
method 3:
```
>> B = [1:4];
>> C = [4:7];
>> A = [B; C];
```
method 4: rand(10,8); % This creates a 10x8 matrix filled with random numbers [0,1]

7. Address matrix elements

(1) A(5, 4)
(2) A(1:2, 2,3)
(3) A(:, 2:3)
(4) A(: , :)
(5) A(2, :)

8. Matrix Operations

(1) matrix added by a scalar

```
>>  A=[1, 2, 3; 3, 4, 5];
>> B = A + 2.0;
 B =
    3     4     5
    5     6     7
```

(2) matrix multiplied by a scalar
(3) matrix addition
(4) element-by-element operation (dot operator)

```
>> A = [1, 2, 3; 4, 5, 6];
>> A.*A
 ans =
        1     4     9
       16    25    36
```

(5) Sum of matrix

```
>> sum(A)
 ans =
        14       77
```

9. Iterations

(1) syntax:

```
for i=1:10
    do something
end
```

Example 1:

```
A = [1:0.2:1.5; 05:-0.1: 0.3]
for myVar = 0.2:0.5:1.3
    A = A * myVar;
end
```

Example 2:

Considering the following iteration code, what is A(12)?

```
A(9)=13;
for i=[10:12] ;
    A(i)=A(i-1)+37;
end
```

10. Nested for-loops

```
for i=1:100
    for j =i:100
     do something
    end
end
```

Example (1): Using nested for-loops to create/modify matrix elements

11 12 13 14 15
13 14 15 16 17
15 16 17 18 19

Matlab code:

```
for i=1:3
    for j = 1:5
        A(i,j) = 8+i *2+j
    end
end
```

11. Plot

Steps to make a plot:
step 1: Guess the plot range, a, b. (-10, 10)
step 2: create x and y vectors. Use x as an independent variable, and use y as a dependent vector: x =-10:0.01:10
y = x.^3 -1
(note: dot operators)
step 3: make a plot: plot(x, y)
Example: Is the following Matlab statement correct or not?

plot([2,3,4, 5], [2,3,4,5]);

You can also plot multiple curves using plot.

12. Fill

Use 'fill' to generate polygons filled with colors.
fill (x, y, 'ok')

Example:

```
x = [0, 0, 1, 1 ];
y = [1, 2, 2, 1 ];
c =[1, 0, 0 ];
fill (x, y, c);
%note: the coordinates of x and y must match. The sequence of the points are important.
```

13. Relational operators

(1) Six relational operators: >, <, >=, <=, ==, ~=
(2) Relational operators can be applied to vectors:

Example:

Given arrays ***x*** = [0 7 3 5] and ***y*** = [2 8 7 0], these are some possible relational operations:

Operation:	**Result:**
k = x<y	k = [1 1 1 0]

```
k = x <= y          k = [1  1  1  0]
k = x == y          k = [0  0  0  0]
```

14. Logic operators

(1) Three logic operators: &, |, ~

(2) Logic operators can be applied to vectors or matices.

Example:

Given arrays ***x*** = [0 7 3 5] and ***y*** = [2 8 7 0], these are some possible relational operations:

Operation:	**Result:**
k = x & y	k = [0 1 1 0]
k = x | y	k = [1 1 1 1]
k = x & ~y	k = [0 0 0 1]

Example:

```
M =[0.2, 0.3, 0.7; 0.6, 0.5, 0.5];
~(M>0.5)
ans:
```

15. if-statements

Syntax:

```
if  statement is true
        do something
elseif
        do anything else
else
        do something
end
```

Or:

```
if  statement
        do something
end
```

Example: Write a Matlab code to find the number of matrix elements greater than a threshold value.

```
M = rand(100);
threshold = 0.25;
counter = 0;
for i=1:100
    for j = 1:100
       if  M(i,j) > threshold
          counter = counter +1;
       end
    end
end
```

16. Colors

RGB

CYMK
[1,1,0]; [1,0,1]; [0 1, 1,], [0, 0, 0]

17. Matrices as images:
(1) Indexed images

18. How to create a image
(1) Create a colormap matrix: (the dimension is n x 3, n colors with each color represented by r, g, b components in 0-1).
(2) Create a matrix: M=rand(50); Designate/alter matrix elements
(3) imagesc(M) and/or imagesc

19. Numerical Integration

(a) integrate the area enclosed by an irregular shape in an image.
(1) use imread (syntax) to load the sample. e.g.,
M = imread('http://cds130.org/wiki/images/Area.jpg', 'jpg');
(2) If the image is with the true-color type, convert it to the indexed image.
new_M = rgb2ind(M, 2)
(note: the new matrix will only contain 0's and 1's. In other words, there are only two colors)
(3) Check the type and size of the image.
[m,n] = size(new_M)

(4) Access the matrix, and identify the areas of interest. (Calculate the total number of 1's and o's).

```
total_area = 0
black_area = 0
for i=0: length(new_M(:,1))    %alternatively you can use [m,n]= size(new_M) to find the size
    for j=0: length(new_M(1, :))
        total_area = total_area + 1;
        if new_M(i, j) > 1
            black_area = black_area +1
        end
    end
end
display('The fraction of the enclosed area is: ');
black_area/total_area;
```

(b) Area underneath a curve.

$$f(x)=x\sin(x^2)+x^2 \quad (\quad 0.2<x<1.3 \quad)$$

Step 1: plot the curve to get a feel of the problem.

```
x = -10:0.01:10;
y = x.*sin(x.^2) + x.^2;
 plot(x,y)
```

Step: 2: discretize the range of x

```
clear;
dx = 0.01;
xmin = 0.2;
xmax = 1.3;
```

Step 3: integrate the area.

```
area=0  % this needs to be initialized before you can use it.

for x=1: (xmax-xmin)/dx
    b1 = x(i)*sin(x(i)^2)+x(i)^2;
    b2 = x(i+1)*sin(x(i+1)^2)+x(i+1)^2;
    trapezoid = 0.5 * (b1 + b2) * dx;
   area = area + trapezoid;
end
```

Alternatively you can try this

Step 3: integrate the area.

```
area=0  % this needs to be initialized before you can use it.

for x=xmin: dx: (xmax-dx)
    b1 = x*sin(x^2)+x^2;
    x = x + dx;
    b2 = x*sin(x^2)+x^2;
    trapezoid = 0.5 * (b1 + b2) * dx;
   area = area + trapezoid;
end
```

Step 4: print out the area that you have integrated.

```
display (area);
```

22 Logic gates.

1) AND, OR, NAND, NOR, XOR, XNOR, NOT gates
2) Truth tables for the logic gates; graphic representation, boolean algebra
3) Build a logic gate using universal logic gates (NAND)
4) Equivalent logic gates